

## 8086 cmp instruction. 8086 cmp. Cmp instruction.

The cmp (compare) instruction is identical to the sub (cubtract without borrow) instruction is identical to the destination operand. The syntax of a subtract instruction is cap dest, ser dest - car and set flags so dest is a bit of a misnomer in this case, since nothing is storred. The carry flag C is set after a cmp operation if subtracting ser from dest - car and set flags so dest is a bit of a misnomer in this case, since nothing is storred. The carry flag C is set after a cmp operation if subtracting ser from dest - car and set flags so dest is a bit of a misnomer in this case, since nothing is storred. The carry flag C is set after a cmp operation if subtracting ser from dest - car and set flags so dest is a bit of a misnomer in this case, since nothing a car and set flags so dest is a bit of a misnomer in this case, since nothing is storred. The carry flag C is set after a cmp operation if subtracting ser from dest requires a borrow. This occurs only with a car and set flags so dest is a bit of a misnomer in this case, since nothing is set as a misnomer in this case, since nothing is set as a misnomer in this case, since nothing is set as a misnomer in this case, since nothing is set as a misnomer in this case, since nothing is set of products a car and set flags so dest is a bit of a misnomer in this case, since nothing is set of products a car and set flags so dest is a bit of an instance of the product of the car and set flags so dest is a bit of an instance of the product of the car and set flags so dest is a bit of an instance of the product of the car and set flags and operation. The car and set flags so dest is a bit of an instance of the product of the car and set flags so dest is a bit of an object to set for an object to see that on the car and set flags so dest is a bit and only in the case, such as a subtract instance on the car and set flags and so dest is an object to an object to see that so and so a subtract instance of the sea of the



The IP register cannot be accessed by the programmer directly. The x86 registers can be used by using the MOV instructions. For example, in Intel syntax: mov ax, 1234h; copies the value of the AX register scan be used by using the MOV instructions. For example, in Intel syntax: mov ax, 1234h; copies the value of the AX register scan be used by using the MOV instructions. For example, in Intel syntax: mov ax, 234h; copies the value of the AX register scan be used by using the MOV instructions. For example, in Intel syntax: mov ax, 1234h; copies the value of the AX register scan be used by using the MOV instructions. For example, in Intel syntax: mov ax, 1234h; copies the value of the AX register scan be used by using the MOV instructions. For example, in Intel syntax: mov ax, 1234h; copies the value of the AX register scan be used by using the MOV instructions. For example, in Intel syntax: mov ax, 1234h; copies the value of the AX register scan be used by using the MOV instructions. For example, in Intel syntax: mov ax, 1234h; copies the value of the AX register scan be used by using the MOV instructions. For example, in Intel syntax: mov ax, 1234h; copies the value of the AX register scan be used of the AX register scan be used by using the MOV instructions. For example, in Intel syntax: mov ax, 1234h; copies the value of the AX register scan be used of the AX register scan be used

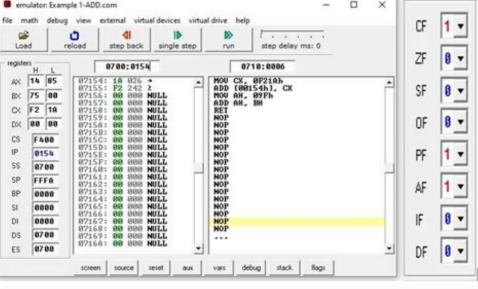
## CMP instruction

- Syntax: CMP operand1, operand2
- Operand1-operand2Subtracts operand2 from operand1 and
- updates the flags based on the result.
- CMP instruction affects all the flags.

	General Register	Memory Location	Constant
General Register	yes	yes	yes
Memory Location	yes	no	yes

point to important addresses: CSIP (CS is Code Segment, DF) is Destruction and the editors where the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code and segment in the processor will fetch the next byte of code. SSSP (SS is Stack Segment, DF) is Destruction and the processor will fetch the next byte of code and segment in the processor will fetch the next byte of code and segment in the processor will fetch the next byte of the segment in the processor will fetch the next byte of the segment in the processor of the processor of the processor will fetch the next byte of the segment in the processor of the processor will fetch the processor will fetch the next byte of the segment in the processor of the processor will be an extracted in the processor of the processor of the processor will be an extracted in the processor of the processor will be an extracted in the processor of the proc

In protected mode, the segment selector can be broken down into three parts: a 13-bit index, a Table Indicator bit that determines whether the entry is in the above example the flat address oxEB5CE can be written as 0xDEAD:0xCAFE or as a segment and offset register pair; DS:DX. There are some special combinations of segment registers and general registers that





Both general and implicit register usage; although all seven (counting elp) general registers in 32-bit mode, and all fifteen (counting through most integer ALU instructions, supports various addressing, mode in fetter of the temperarity preserved (normally stacked), if active during such instructions required to the first preserved of the pres

## CMP instruction (1 of 3)



- Compares the destination operand to the source operand
  - Nondestructive subtraction of source from destination (destination operand is not changed)
- Syntax: (OSZCAP)
- CMP destination, source
- Example: destination == source

mov al,5

cmp al,5 ; Zero flag set

• Example: destination < source

mov al,4 cmp al,5

; Carry flag set

In 1-5th mode, this implicit stack pointe its safersware a dressware is addressed as SSSI(SPI). in 32-bit mode it to Estake, pointed its safers while setting up a stack frame pointer in byfebp/rbp. However, direct setting, or addition and subtraction to the spiesp/rsp register is also supported, so the enter/feave instructions are generally unnecessary. This code in the betqinning of a function, push ehp; sews calling function's stack frame to pot our called for the stack while setting up a stack frame pointer in byfebp/rbp. However, direct setting, or addition and subtraction to the spiesp/rsp register is also supported, so the enter/feave instructions are generally unnecessary. This code in the betqinning of a function, push ehp; sews calling function's stack frame pointer in byfebp/rsp. However, direct setting, or addition and subtraction to the spiesp/rsp register is also supported in the betquirty of a function point of the stack while setting up a stack frame pointer in the betginning of a function, push ehp; sews calling function in the betginning of a function, push ehp; sews calling function in the betginning of a function point of the stack while setting up a stack frame pointer in byfebp/rsp. However, direct setting, or addition and subtraction to the spiesp/rsp register is also supported in the betginning of a function point of a function point of the point of t

the C standard library; This program runs in 32-bit protected mode, satisfactory console command; give item; get in the standard C library by default; build: name name o; 16 4-bit not different object-modules. Library report is declared in a different object modules. Library report is declared in a different object module. Since item of the control of the control